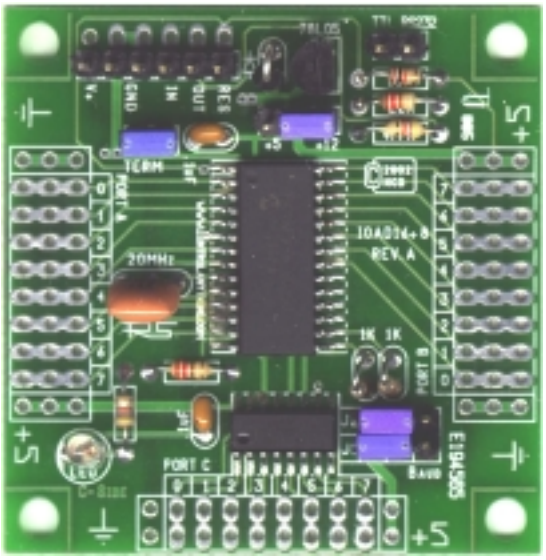
IOAD168 I/O Controller

RS-232 E3C Networkable Multi-Function I/O A/D Controller



The IOAD168 is a multi-function Digital Input/Output controller with a 5-channel 8 or 10-Bit A/D converter. The IOAD168 connects to the serial port of any computer and responds to commands from the user at baud rates up to 38.4K.

The IOAD168 was designed to offer desktops, laptops, and embedded computers access to low-level digital I/O and A/D functions with minimal programming. Visual Basic program examples are provided in this manual and are available for download at ***www.controlanything.com***.

The IOAD168 supports commands for setting the on/off state of individual lines, setting the state of all lines at once (serial-to-parallel conversion), reading the state of I/O lines (single input or parallel-to-serial encoding), reading analog values, and mixing inputs and outputs on a single I/O data port. In addition, the IOAD168 incorporates a dedicated 8-bit output-only data port. The IOAD168 is E3C compliant, allowing 256 NCD devices to be attached to a single RS-232 serial port in any combination. The IOAD168 is also firmware upgradeable, allowing for future upgrades to add new features at no additional cost.

The IOAD168 is compatible with many expansion modules (available separately) for adding I/O functions to the controller. See below for current compatibility list.

Control 256 Devices from a Single Serial Port
User-Selectable Communication Rates from 2400, 9600, 19.2K, and 38.4K Baud
E3C Compliant Command Set
Device Enabled/Power LED
+5 or +12 Volt DC Operation
Dual 8-Bit I/O Data Port + 8-Bit Output Only Port
5 Channel 8-Bit or 10-Bit Software Selectable A/D Converter
O.C. RS-232 Communication for Networking Multiple Devices
Powerful ASCII Character Code Based Command Set
Compatible with ANY Computer or Microcontroller
Expansion Module Upgradeable
Firmware Upgradeable FLASH Memory

Compatible Expansion Modules

8FET High Power 8-Channel FET Driver
Useful for Direct Control of Relays, Stepper Motors, or Other High-Power Devices. Connect one of these to the IO Data bus on any controller, or Connect 3 of these to a single PAR24 Expansion Module. Compatible with All 3 8-bit Data Ports on the IOAD168.

AD1216 12-Bit 16-Channel Analog to Digital Converter Expansion Module
Up to 3 AD1216s can share a Single IO Data Bus. Connect up to 6 AD1216s on a Single IOAD168 controller, allowing software monitoring of 96 analog data channels. Not compatible with the Output Only port on the IOAD168.

DCM1 High Power DC Motor Controller
Includes Forward, Reverse, Braking, and 250+ Levels of Speed Control. Connect up to 3 of these to a single IOAD168, or Connect 3 of these to a single PAR24 Expansion Module.

PAR24 24-Bit Parallel Output Expansion Module
Each PAR24 Adds 24 TTL Outputs to the IO Data Bus. Connect up to 4 of these to a single IOAD168 controller. Not compatible with the Output Only port on the IOAD168.

TRDVR Low Power 8-Channel O.C. Transistor Driver
Useful for Direct Control of small Relays, LEDs, or other low current devices. Connect up to three of these to the IOAD168, or Connect 3 of these to a single PAR24 Expansion Module.

TESTL 10-LED Output Test Module
Connect up to three of these to the IOAD168, or Connect 3 of these to a single PAR24 Expansion Module.

TESTD 8-DIP Switch Input Test Module

5-Year Repair or Replace Warranty

Warranty
NCD Warrants its products against defects in materials and workmanship for a period of 5 years. If you discover a defect, NCD will, at its option, repair, replace, or refund the purchase price. Simply return the product with a description of the problem and a copy of your invoice (if you do not have your invoice, please include your name and telephone number). We will return your product, or its replacement, using the same shipping method used to ship the product to NCD.

This warranty does not apply if the product has been modified or damaged by accident, abuse, or misuse.

30-Day Money-Back Guarantee
If, within 30 days of having received your product, you find that it does not suit your needs, you may return it for a refund. NCD will refund the purchase price of the product, excluding shipping/handling costs. This guarantee does not apply if the product has been altered or damaged.

Copyrights and Trademarks

Disclaimer of Liability
NCD is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs or recovering, reprogramming, or reproducing any data stored in or used with NCD products.

Technical Assistance
Technical questions should be e-mailed to Ryan Sheldon at ncdryan@aol.com. Technical questions submitted via e-mail are answered up to 20 times daily. Technical support is also available by calling (417) 646-5644.

NCD Contact Information

Mailing Address:
National Control Devices
P.O. Box 455
Osceola, MO 64776

Telephone:
(417) 646-5644

FAX:
(417) 646-8302

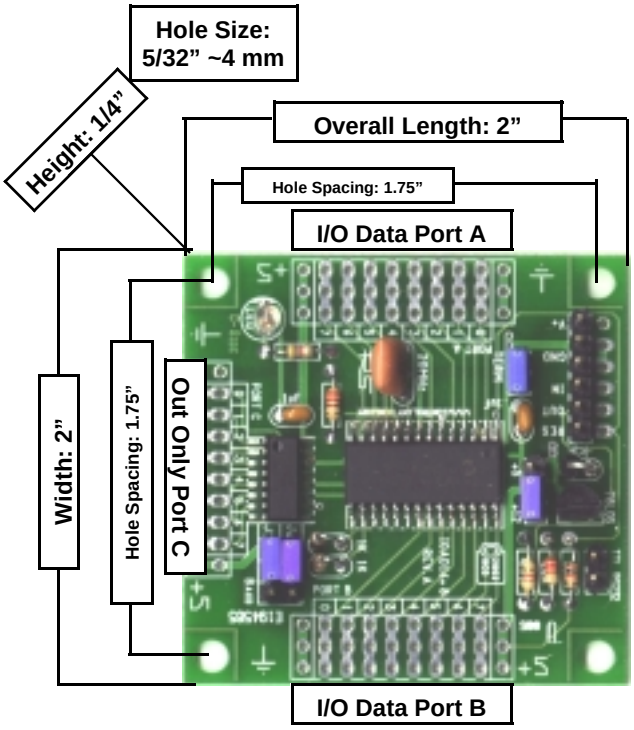
Internet:
ryan@controlanything.com
www.controlanything.com
www.controleverything.com

IMPORTANT POWER SUPPLY REQUIREMENTS

1) DO NOT USE A WALL WART TYPE UNREGULATED POWER SUPPLY.
2) USE ONLY A COMPUTER GRADE REGULATED SWITCHER SUPPLY RATED AT +5 or 12 VOLTS DC, 200 ma OR GREATER.
3) USE A SUPPLY RATED FOR MORE AMPERAGE WHEN POWERING MULTIPLE BOARDS.
4) DC POWER SHOULD NEVER TRAVEL GREATER THAN 20 FEET. A SEPARATE POWER SUPPLY SHOULD BE USED FOR EACH CONTROLLER IF CONTROLLERS ARE NOT LOCATED WITHIN 20 FEET OF EACH OTHER.
5) THE IOAD168 CAN BE USED IN 12 VOLT AUTOMOTIVE ELECTICAL SYSTEMS WHEN SET TO +12 VOLT MODE. SOME EXPANSION MODULES SHOULD NOT BE USED WHEN SET TO +12 VOLTS.

Status LED:

When power is first applied to the IOAD168, the power LED will glow brightly. If an E3C command is used to disable the IOAD168, the power LED will glow dimly. If the LED is not on at all, the IOAD168 is not getting any power. If the LED glows dimly when power is first applied to the board, the IOAD168 CPU is damaged.



+5 or +12 Volt Input
Power Ground
RS-232 Ground
RS-232 Data Input
RS-232 Data Output
Reset

Power and Data Interface

+5 or 12 Volt Input
This pin supplies power to the board. ***MAKE SURE THE POWER JUMPER IS SET PROPERLY OR PERMANENT DAMAGE WILL RESULT***.

Power Ground
RS-232 Ground
These ground lines are tied together on the IOAD168 board.

RS-232 Data Input
Connect this line to the RS-232 Data OUTPUT of your Computer or Microcontroller.

RS-232 Data Output
Connect this line to the RS-232 Data INPUT of your Computer or Microcontroller

RESET
Connect this line to ground to reboot the on-board microcontroller.

RS-232 Communications

User Selected Baud Rate
8 Data Bits
1 Stop Bit
No Parity

Jumper Settings

TERM
The RS-232 Data Output of the IOAD168 is Open Collector and MUST be terminated. Install the TERM jumper if you are only connecting 1 IOAD168 to your serial port. If you are using multiple IOAD168 boards on a single serial port, you should remove the TERM jumper on all additional boards. Leave the TERM jumper installed on the IOAD168 that is wired closest to the computer.

+5/+12 Voltage Select
Make sure this jumper is set properly for the voltage level you are using. Improper setting of this jumper will damage the IOAD168. If you purchased a Quick Start Kit with the IOAD168, this jumper must be set to +5.

TTL RS-232
Install a jumper across these pins if the RS-232 data source is at TTL logic levels

Baud 1 & 2:
These Jumpers select the baud rate for the IOAD168. The photo shows the baud rate set at 38.4K baud. Install the jumpers as follows for other baud rates:

2400 Baud	Set Jumpers J1 and J2 in the Down position.
9600 Baud	Set Jumper J1 to the Down Position, Set Jumper J2 in the Up Position.
19.2K Baud	Set Jumper J1 to the Up Position, Set Jumper J2 to the Down Position.
38.4K Baud	Set Both Jumpers in the UP as shown in the photo above.

Chaining Multiple IOAD168s on a Single RS-232 Serial Port

Step 1:
Connect Each IOAD168 to your computer by itself and program each board with a unique device number from 0-255.

Step 2:
It would be helpful to use an NCD Quick Start Kit (Part#: QS5-F6) to Apply Power and Data to the First IOAD168.

Step 3:
Connect the V+, GND, In and OUT lines (and optionally the RESET line) of the first board to the V+, GND, In and OUT lines of next board. Keep chaining IOAD168s together in this manner until you have enough controllers attached to meet your needs.

Step 4:
Remove the TERM jumper on all IOAD168 boards except for the board that is wired closest to your computer.

Step 5:
Use E3C command 252 to select an IOAD168 board to control using the programmed device number. All subsequent commands will be seen only by the board you have selected with this command. Use command 252 again to switch to a different board.

Two-Way Communication:

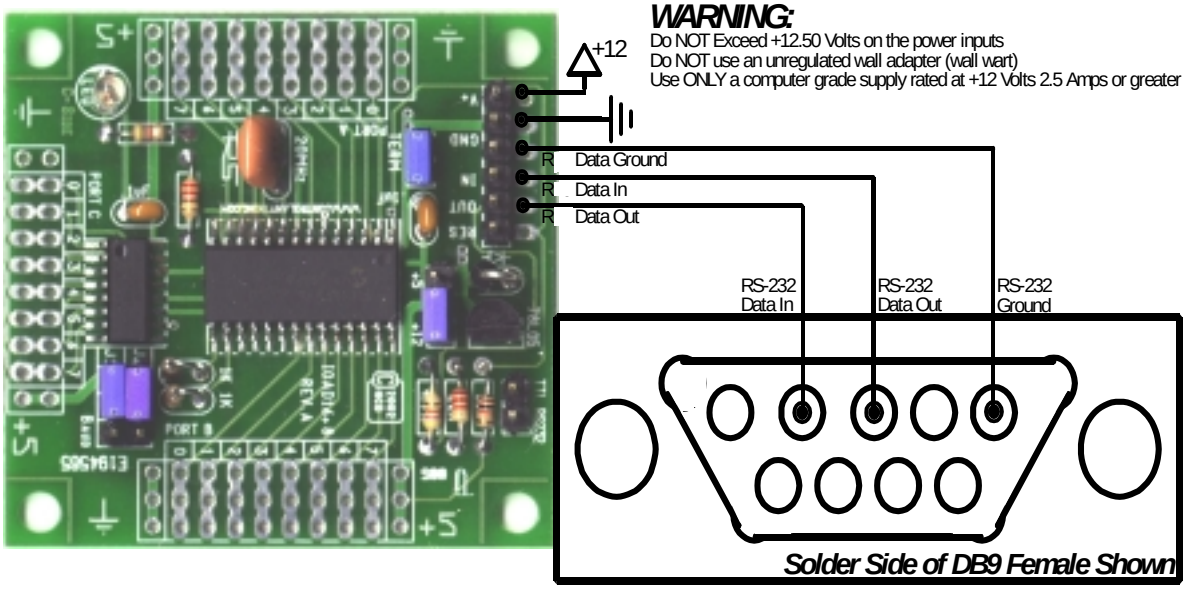
The IOAD168 supports two-way communication for confirming the receipt of commands and for reporting the status of IO Lines back to the host computer.

The IOAD168 should be connected as shown below when using this device for the first time. Even if you plan to connect several IOAD168 controllers to a single serial port, this wiring diagram must first be used to program a unique device number into the controller.

One-Way Communication:

The Feature Set of the IOAD168 is very powerful, but if you do not need to retrieve any data from the board, it is possible to use the IOAD168 via one-way communications. Simply omit the connection of the Data Out line of the IOAD168 to the RS-232 Data In line on the computer. In this mode, the IOAD168 can be controlled from an RS-232 serial port using just 2 wires over great distances, but functions are limited to output-only commands.

Two-Way Communication Wiring:



Sending Commands to the IOAD168

The IOAD168 is capable of sending and receiving data via RS-232 serial communications. The IOAD168 is compatible with just about any computer or microcontroller ever produced, including the Macintosh, Amiga, Basic Stamp, and of course, Windows & DOS based machines.

Regardless of the system you are using, you will need access to a programming language that supports program control of the serial port on your system.

A terminal program is not suitable for controlling the IOAD168 . Commands should be sent using ASCII character codes 0-255 rather than ASCII characters (A, B, C etc.). See "ASCII Codes vs. Characters" on this page.

Most systems require you to open the appropriate serial port (COM port) prior to sending or receiving data.

Because there are so many different ways to send and receive data from various languages on various platforms, we will provide generic instructions that can be easily converted to your favorite language.

For example, if this manual says "Send ASCII 254", the user will need to translate this instruction into a command that is capable of sending ASCII character code 254.

To Send ASCII 254 from Visual Basic, you will use the following line:

MSComm1.Output = Chr$(254)

In Qbasic, you can send ASCII 254 using the following line of code:

Print #1, Chr$(254);

Note that sending ASCII character code 254 is NOT the same as sending ASCII characters 2, 5, and 4 from a terminal program. Typing 2, 5, and 4 on the keyboard will transmit three ASCII character codes.

In your program, you may want to ask the IOAD168 for the current status of the relays, just to confirm their activation. If so, your programming language will support commands for reading data from the serial port.

For your convenience, we have provided several programming examples in Visual Basic 6 for controlling the R4x/R8x Pro. These examples should greatly speed development time. You may want to visit **www.controleverything.com** for the latest software and programming examples.

Programming examples for the IOAD168 are much more extensive for Visual Basic 6 users than for any other programming language. If you are not a VB programmer, you may consider looking at the VB6 source code, as it is easily translated into other popular languages.

Regardless of your programming background, the provided Visual Basic 6 source code is very easy to understand and will likely resolve any communication questions you may have. VB6 programming examples may be viewed in any text editor.

ASCII Codes vs. Characters

The differences between ASCII codes and ASCII characters tend to generate a lot of confusion among first-time RS-232 programmers. It is important to understand that a computer only works with numbers. With regard to RS-232 data, the computer is only capable of sending and receiving numbers from 0 to 255.

What confuses people is the simple idea that the numbers 0 to 255 are assigned letters. For instance, the number 65 represents the letter A. The number 66 represents the letter B. Every character (including numbers and punctuation) is assigned a numeric value. This standard of assignments is called ASCII, and is a universal standard adopted by all computers with an RS-232 serial port.

ASCII characters codes can be clearly defined as numbers from 0 to 255.

ASCII characters however are best defined as letters, A, B, C, D, as well as punctuation, !@#$%, and even the numbers 0-9.

Virtually all programming languages permit you to send ASCII in the form of letters or numbers. If you wanted to send the word "Hello" out the serial port, it is much easier to send the letters H, e, l, l, and o than it is to send the ASCII character codes that represent each letter.

For the purposes of controlling NCD devices however, it is much easier to build a numeric command set. Especially when communicating to devices where you want to speak to lots of outputs (which are numbered), inputs (which are also numbered), or control specific devices using their device number (from 0 to 255).

Put simply, it is easier to control NCD devices using ASCII character codes 0 to 255 than it is to use ASCII characters A, B, C, D, etc.

Because terminal programs are ASCII character based, it may be difficult to generate the proper series of keystrokes that would be necessary to activate a particular function. Therefore, they are not suitable for controlling NCD devices. In a real world control application, a terminal program would not likely be used to control NCD devices anyway. Therefore, a programming language that supports the transmission and reception of ASCII character codes 0 to 255 is highly recommended.

The E3C Command Set: Software Control of Multiple NCD Devices

The E3C command set allows you to control up to 256 NCD devices from a single serial port. It is OK to mix different types of devices, as long as the devices are E3C compliant. The IOAD168 supports the full set of E3C commands, plus a set of extended commands for storing and recalling the device number.

How does E3C Work?
First of all, each device must be assigned a device number from 0 to 255. The IOAD168 must be programmed with a device number, which is accomplished using the "Store Device Number" command shown below.

E3C stands for Enabled 3-Wire Communication. Put simply, when you first power up your computer and all the devices attached to the serial port, all devices will respond to your commands.

Using the E3C command set, you can specify which devices will listen and which devices will ignore your commands. Note that E3C commands are never ignored by any device, regardless of the commands you send to the controller.

The number to the left of each command indicates the ASCII character code that must be sent to issue the command. All commands must be preceded with ASCII character code 254 to place the device in command mode. See examples at right.

The E3C Command Set

248 Enable All Devices:
Tells all devices to respond to your commands.

249 Disable All Devices:
Tells all devices to ignore your commands.

250 Enable a Selected Device:
Tells a specific device to listen to your commands.

251 Disable Selected Device:
Tells a specific device to ignore your commands.

252 Enable Selected Device Only:
Tells a specific device to listen to your commands, all other devices will ignore your commands.

253 Disable a Selected Device Only:
Tells a specific device to ignore your commands, all others will listen.

E3C Visual Basic Programming Examples

The E3C command set is easily used from any programming language that supports serial communication. The following Visual Basic 6 Example source code demonstrates subroutines that can be used to control which devices will listen and which devices will ignore your commands.

Most commands issued to the IOAD168 are acknowledged by sending ASCII character code 85 back to the host computer (when reporting is turned on). E3C commands are not acknowledged regardless of the reporting mode.

Sample Code: The E3C Command Set

```
Public Sub EnableAllDevices()
    'Enable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(248)    'E3C Enable All Device Command
End Sub

Public Sub DisableAllDevices()
    'Disable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(249)    'E3C Disable All Device Command
End Sub

Public Sub EnableSpecificDevice(Device)
    'Enable A Specific E3C Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(250)    'E3C Disable Specific Device Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Disabled
End Sub

Public Sub DisableSpecificDevice(Device)
    'Disable A Specific E3C Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(251)    'E3C Disable Specific Device Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Disabled
End Sub

Public Sub DisableAllDevicesExcept(Device)
    'Disable All E3C Devices Except (Device)
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(252)    'E3C Disable All Device Except Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Active
End Sub

Public Sub EnableAllDevicesExcept(Device)
    'Enable All E3C Devices Except (Device)
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(253)    'E3C Enable All Device Except Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Inactive
End Sub
```

The IOAD168 Command Set and Controller Configuration

The IOAD168 supports an extensive command set, used to control IO lines, set operation modes, and store and recall IO status lines. Most users will not use many of the functions built into this controller. The best way to familiarize yourself with the capabilities is to carefully read through the command set in this section. The "plain English" examples provide a quick, easy to understand definition of what each command does.

The number to the left of each command indicates the ASCII character code that must be sent to issue the command. All commands must be preceded with ASCII character code 254 to place the device in command mode. See examples at right.

Visual Basic Programming Examples

A Visual Basic 6 programming examples are provided in the following pages to assist in the development of software for controlling the IOAD168. Additional source code can be found on our web site at www.controlanything.com.

Test 2-Way Communications

0, 0 Test 2-Way Communication

This command can be used to test 2-way communication between the host computer and the relay controller. When executed, the relay controller will send ASCII character code 85 back to the user. This command should be used for initial installations if 2-way communication is required. It can also be used to detect the presence of a relay controller on the serial port.

Sample Code: Test 2-Way Communications

```
Public Function Test2Way
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Mode
    MSComm1.Output = Chr$(0)            'Request 2-way Comm. Test
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    Test2Way = Asc(MSComm1.Input)       'Get Status from Controller
    Debug.Print Test2Way                'Display in Immediate Window
End Sub
```

Default Settings

0, 1 Default Settings

This command can be used to restore all configuration parameters to factory default settings. DO NOT ISSUE THIS COMMAND IF MORE THAN ONE IOAD168 CONTROLLER IS ATTACHED TO THE SAME SERIAL PORT OR ALL CONTROLLERS WILL BE PROGRAMMED WITH FACTORY DEFAULT VALUES. When this command is sent, parameters are changed to the following default values:

Default Keypad Settings

1	**2**	**3**	**A**
4	**5**	**6**	**B**
7	**8**	**9**	**C**
*****	**0**	**#**	**D**

Default PIN Character = X

The IOAD168 has the ability to control character LCD and VF displays. In addition, keystrokes from the Keypad can be sent directly to the display screen. In some cases, such as a PIN or PASSWORD request, the characters should be hidden from other viewers, but acknowledged by the display. The default character used to hide the users input is "X".

Default Character Delay = 150

The ***Character Delay*** value sets the delay between bytes of data sent from the controller to the user. A value of 150 is very safe for slow computers. If you are using a fast computer, two-way communication speed can be significantly increased by lowering the character delay value. However, a value set too low can generate communication errors.

Default E3C Device Number = 0

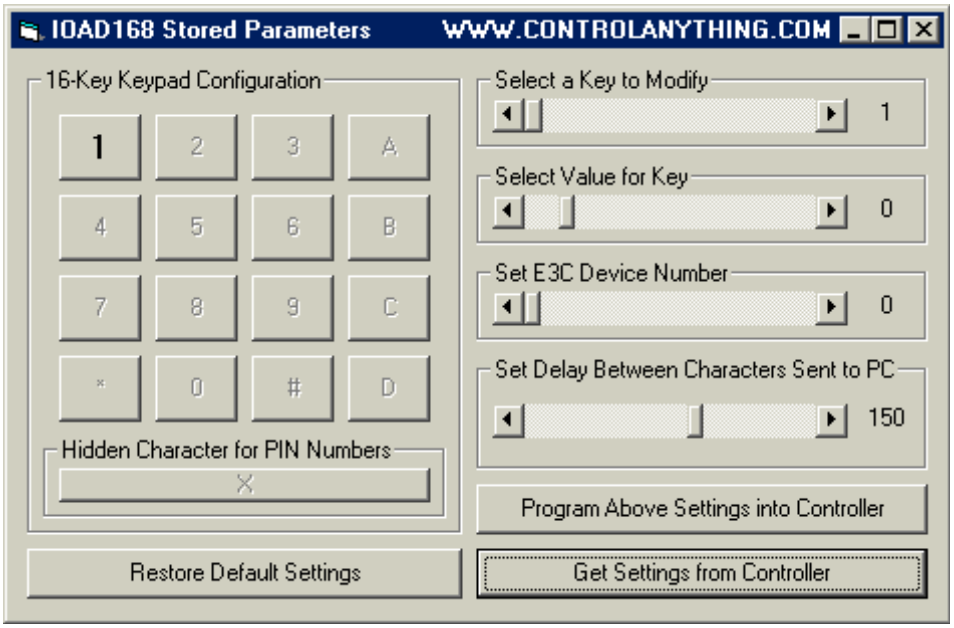
Up to 256 E3C compliant devices can share a single serial port in any combination. The E3C device number is used to define the "location" of the IOAD168 on the E3C network.

When this command has completed programming of default values, the controller will send ASCII character code 85 back to the user. At this point, power should be removed and re-applied to the controller for all changes to take effect. If you are using a Quick Start Kit (Part Number QS5-F6, simply press the reset button to reboot the firmware).

Sample Code: Default Settings

```
Public Sub Default
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Mode
    MSComm1.Output = Chr$(1)            'Request 2-way Comm. Test
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    Test2Way = Asc(MSComm1.Input)       'Get Status from Controller
    Debug.Print Test2Way                'Display in Immediate Window
End Sub
```

The IOAD168 Command Set and Controller Configuration



The IOAD168 can be programmed with several parameters either by the user or by using our compiled IOAD168 configuration program. The easiest way to configure the controller is to use our compiled configuration program, which provides a simple visual interface for permanently configuring the IOAD168 controller.

Maximizing Performance
The "Set Delay Between Characters" slider should be decreased for better performance. High speed compiled programs may even allow for a "0" setting, making communication from controller to PC MUCH FASTER.

Programming Keypad Keys:
Use the "Select a Key to Modify" Slider to choose a button to change. Next, use the "Select Value for Key" slider to change the button. The "Hidden Character" is the character this displayed in place of actual key presses to hide the feedback from other viewers (when used with a character display).

When finished, select "Program Settings into Controller".

Running Compiled Program Examples

Please examine our complete library of software examples for the IOAD168 at www.controlanything.com. The examples shown are code fragments only and require other components to operate properly. The configuration program used to set the parameters of the IOAD168 is provided with complete source code (viewable in any text editor), and as a set of four compiled applications (executable .EXE programs). If you choose to run the COMPILED (.EXE) version, set the IOAD168 controller for 38.4K baud. Then select the application suited for your system (one for each of the four COM ports). If you have difficulty running the compiled programs, go to www.controlanything.com and visit the graphic displays section of our web site. Next, locate, download, and install the file: Image Loader Utility Version 1.01 or newer (ILUV101.ZIP). This will install the necessary components on your system required to run all Visual Basic program examples that we have compiled into .EXE applications.

Programming and Retrieving Configuration

The code fragments below show a general example for storing and retrieving configuration data into and out of the IOAD168. All parameters are stored and retrieved sequentially. Pay attention to the "GetData" functions in these code examples. These functions are used to keep data synchronized between the computer and the controller. Any time a "GetData" function is encountered below, the program will wait for the IOAD168 to send ASCII character code 85 back to the computer. The easiest way to configure the IOAD168 is to use a Windows based computer and a Quick Start kit. Make sure only ONE controller is attached to the serial port when programming parameters or all devices will be programmed with the same configuration data.

Sample Code Fragment: Programming Settings

```
Private Sub ProgramSettings_Click()

    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(0)    'Enter Setup Command
    MSComm1.Output = Chr$(2)    'Program Settings Command
    Debug.Print GetData         'Wait for Command to Begin
    '1 = E3C Device Number
    MSComm1.Output = Chr$(HScroll3.Value)

    GetData                     'Wait for Parameter to Store
    '2 = Character Delay Value (0-255) 255 = slow
    MSComm1.Output = Chr$(CDEL.Value)
    GetData                     'Wait for Parameter to Store
    'Program the Keypad and Hidden PIN Characters
    For n = 0 To 16
       MSComm1.Output = Chr$(Asc(Key(n).Caption))
       For del = 0 To 250: DoEvents: Next del
       GetData                  'Wait for Parameter to Store
    Next n
End Sub

Public Function GetData()
       t = 0
        Do
            t = t + 1
            If t > 10000 Then GoTo ext

            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        GetData = Asc(MSComm1.Input)
        Debug.Print GetData
        Exit Function
ext:
        GetData = 88
End Function
```

Sample Code Fragment: Retrieve Settings

```
Public Sub GetSettings_Click()
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(0)            'Enter Setup Command
    MSComm1.Output = Chr$(3)            'Get Settings Command
    Debug.Print "Waiting..."
    If GetData = 254 Then               'Begin Retrieval of Settings
        HScroll3.Value = GetData        '1 = E3C Device Number
        CDEL.Value = GetData            '2 = Character Delay
        For n = 0 To 16                 'Get Keypad Character Codes
            Key(n).Caption = Chr$(GetData)
            'MSComm1.Output = Chr$(Asc(Key(n).Caption))
            GetData
        Next n
        Debug.Print GetData
    End If
End Sub

Public Function GetData()
       t = 0
        Do
            t = t + 1
            If t > 10000 Then GoTo ext

            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        GetData = Asc(MSComm1.Input)
        Debug.Print GetData
        Exit Function
ext:
        GetData = 88
End Function
```

The IOAD168 Command Set

Sending a Byte of Data to a Data Port

1, 0-255 Put Byte on Port A

This command is used to send a byte of data directly to the Port A data port on the IOAD168. All lines on Port A become an output when this command is issued. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

2, 0-255 Put Byte on Port B

This command is used to send a byte of data directly to the Port B data port on the IOAD168. All lines on Port B become an output when this command is issued. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

3, 0-255 Put Byte on Port C

This command is used to send a byte of data directly to the Port C data port on the IOAD168. Port C is an Output-Only port. This command is very similar to a serial-to-parallel converter. The parameter of this command, 0-255, is written directly to the port and the on/off status of each of the 8 lines appears in the equivalent binary pattern of the parameter.

Sample Code: Send Byte to Port

```
Public Sub PortAByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(1)            'Send Data Byte to Port A
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub

Public Sub PortBByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(2)            'Send Data Byte to Port B
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub

Public Sub PortCByte(DatByte)           'DatByte Parameter = 0 to 255
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(3)            'Send Data Byte to Port C
    MSComm1.Output = Chr$(DatByte)      'Data Byte to Appear on Port
End Sub
```

Setting Individual Data Bits

4, 0-47 Set Port Bits

This command is used to set the on/off status of individual I/O lines on each of the 3 data ports. This command requires a parameter of 0-47.
Parameter Values 0-7 Turn Off Port A Bits
Parameter Values 8-15 Turn On Port A Bits
Parameter Values 16-23 Turn Off Port B Bits
Parameter Values 24-31 Turn On Port B Bits
Parameter Values 32-39 Turn Off Port C Bits
Parameter Values 40-47 Turn On Port C Bits

Sample Code: Setting Status of Port Bits

```
Public Sub PortBitSet(BitSet)           'BitSet Parameter = 0 to 47
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(4)            'Send BitSet Command
    MSComm1.Output = Chr$(BitSet)       'Set Bit Status
End Sub
```

Reading Port Bits

5, 0-23 Get Status of Port Bits

This command is used to read the status of data bits on each of the 3 data ports. This command requires a parameter value of 0-23. This command returns an ASCII character code 0 or 1 indicating bit status.
Parameter Values 0-7 Set Bit to Input and Read Port A Data Bit
Parameter Values 8-15 Set Bit to Input and Read Port B Data Bit
Parameter Values 16-23 Read Port C Bits, Cannot Be Used as Input

Sample Code: Read Port Bit Status

```
Public Function PortBitGet(GetBit)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(5)            'Request Status of Port Bit
    MSComm1.Output = Chr$(GetBit)       'Port Bit to Read
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetBit = Asc(MSComm1.Input)         'Get Status from Controller
End Sub
```

The IOAD168 Command Set

Reading Port Bytes

The IOAD168 is capable of reading data from each of the various data ports. The Read Port Byte command is rather extensive because it is used to report any data generated by the IOAD168 back to the user. These commands include reading I/O status information, analog values, and data generated by various expansion modules. For this reason, each parameter for the Read Port Byte command will be thoroughly explained.

Read Port Byte from Data Port

6, 0 Read Port A Data Byte
This command is used to read a binary value from Port A, acting like a parallel-to-serial encoder. This command switching all data bits on Port A to an input, takes a reading, and reports a value of 0-255 back to the user.

6, 1 Read Port B Data Byte
This command is used to read a binary value from Port B, acting like a parallel-to-serial encoder. This command switching all data bits on Port B to an input, takes a reading, and reports a value of 0-255 back to the user.

6, 2 Read Port C Data Byte
This command is used to report the current output status of Port C. Port C cannot be used to read inputs, therefore, it can only tell you what the current output status is currently set to. This command reports a value of 0-255 back to the user.

Sample Code: Read Port Byte

```
Public Function GetPortAStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(0)            'Request Port A Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub

Public Function GetPortBStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(1)            'Request Port B Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub

Public Function GetPortCStatus
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request a Port Byte
    MSComm1.Output = Chr$(2)            'Request Port C Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetPortAStatus = Asc(MSComm1.Input) 'Get Status from Controller
End Sub
```

Reading 8-Bit Analog Values

6, 3-7 Read 8-Bit Analog Data on Channels 1-5
The first 5 lines (labeled 0-4 on IOAD168 circuit board) on Port A can be used to read 8-bit analog values from a variable 0-5VDC data source. This command requests the analog data value from the user-specified A/D input, and reports back an analog value of 0-255.

Sample Code: Reading 8-Bit Analog Values

```
Public Function GetAnalog8(Channel)     'Channel = 0 to 4
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request 8-Bit Analog Value
    MSComm1.Output = Chr$(3+Channel)    'Request from Channels 0-4
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
```

Reading 10-Bit Analog Values

6, 8-12 Read 10-Bit Analog Data on Channels 1-5
The first 5 lines (labeled 0-4 on IOAD168 circuit board) on Port A can be used to read 10-bit analog values from a variable 0-5VDC data source. This command requests the analog data value from the user-specified A/D input, and reports back two bytes that are computed to indicate an analog value from 0-2047.

Sample Code: Reading 10-Bit Analog Values

```
    GetAnalog8 = Asc(MSComm1.Input)     'Get Analog Value from IOAD168
End Sub
Public Function GetAnalog8(Channel)     'Channel = 0 to 4
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(6)            'Request 10-Bit Analog Value
    MSComm1.Output = Chr$(8+Channel)    'Request from Channels 0-4
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    MSB = Asc(MSComm1.Input)            'Get Most Significant Byte
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    LSB = Asc(MSComm1.Input)            'Get Least Significant Byte
    GetAnalog10 = (MSB*256)+LSB         'Compute Analog Value
End Sub
```

The IOAD168 Command Set

Expansion Modules

The IOAD168 is capable of interfacing to a number of 3rd party products and expansion modules. This section demonstrates the use of the IOAD168 with various other products such as Keypads, LCD Displays, and of course, NCD Expansion Modules.

PAR24: 24-Bit Parallel Output

The PAR24 is a 24-Bit Parallel Output Expansion module that can be connected to Port A or Port B of the IOAD168 controller. The PAR24 provides three sets of 8 TTL/CMOS compatible outputs. Each set of eight outputs is called a “Channel”. If you are using ONE PAR24 connected to PORTA, then channels 1-3 are available. If you are using TWO PAR24s connected to PORTA, channels 1-6 are available. Up to FOUR PAR24 expansion modules can be connected to a single IOAD168 controller. Two on Port A and two on Port B. This will provide 96 TTL/CMOS outputs + 8 outputs on Port C of the IOAD168. The following commands are used to send bytes of data to the outputs of PAR24. A single byte of data is easily routed to the user selected Port and Channel using the integrated PAR24 command set.

PAR24: PORT A and PORT B

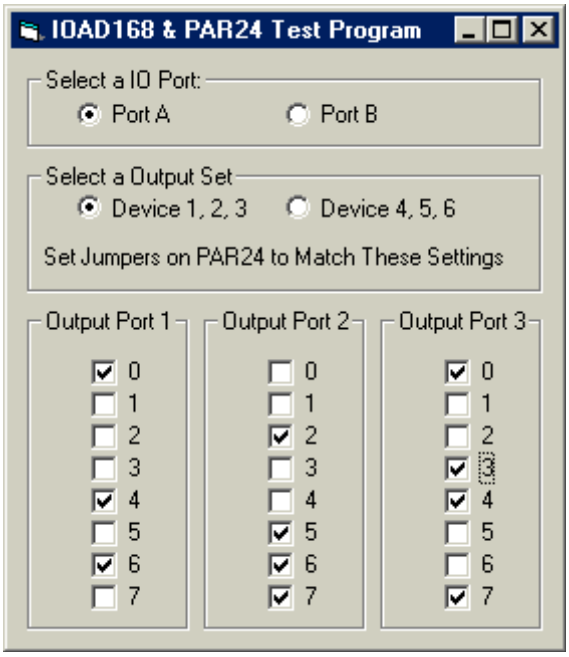
11, 1-6, 0-255 PAR24 Output Byte PortA
This command sends a byte of data (0-255) to one of the PAR24 output channels (1-6) on Port A (Command 11). The data appears as binary/parallel data on the 8-bit output of the PAR24.

12, 1-6, 0-255 PAR24 Output Byte PortA
This command sends a byte of data (0-255) to one of the PAR24 output channels (1-6) on Port B (Command 12). The data appears as binary/parallel data on the 8-bit output of the PAR24.

Sample Code: PAR24

```
Public Sub Par24A(Channel,Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(11)           'PAR24 on Port A Command
    MSComm1.Output = Chr$(Channel)      'Select Output Channel 1-6
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
End Sub

Public Sub Par24B(Channel,Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(12)           'PAR24 on Port B Command
    MSComm1.Output = Chr$(Channel)      'Select Output Channel 1-6
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
End Sub
```



We also provide a PAR24 demo program in the form of Visual Basic 6 Source Code at www.controlanything.com.

The IOAD168 Command Set

AD1216: 16-Channel 12-Bit A/D Converter

The AD1216 is an expansion module with a 16-Channel 12-Bit A/D Converter that is compatible with Port A or Port B on the IOAD168 controller. Up to Six AD1216s can be connected to a single IOAD168, three on Port A and three on Port B. This allows monitoring of 96 analog data sources at 12-bit resolution. The AD1216 command set makes it very easy to request the analog value on any port, device, and channel. Two bytes of data are sent back to the user after a successful A/D conversion. The Least Significant Byte (LSB) is sent first, followed by the Most Significant Byte (MSB). To convert the MSB and LSB to an analog value of 0-4095, use the following equation:

ANALOG_VALUE = (MSB x 256) + LSB

HINT: You can increase the samples per second far above what is shown in the picture below by setting the baud rate to 38.4K baud and by decreasing the delay between characters. A fast computer is required. See Page 8 for Details.

6, 13, 0-2, 0-15 Get 12-Bit Analog on Port A
This command returns a 12-Bit analog value from Channel 0-15, Defined as Device 0-2 on a AD1216 Connected to Port A.

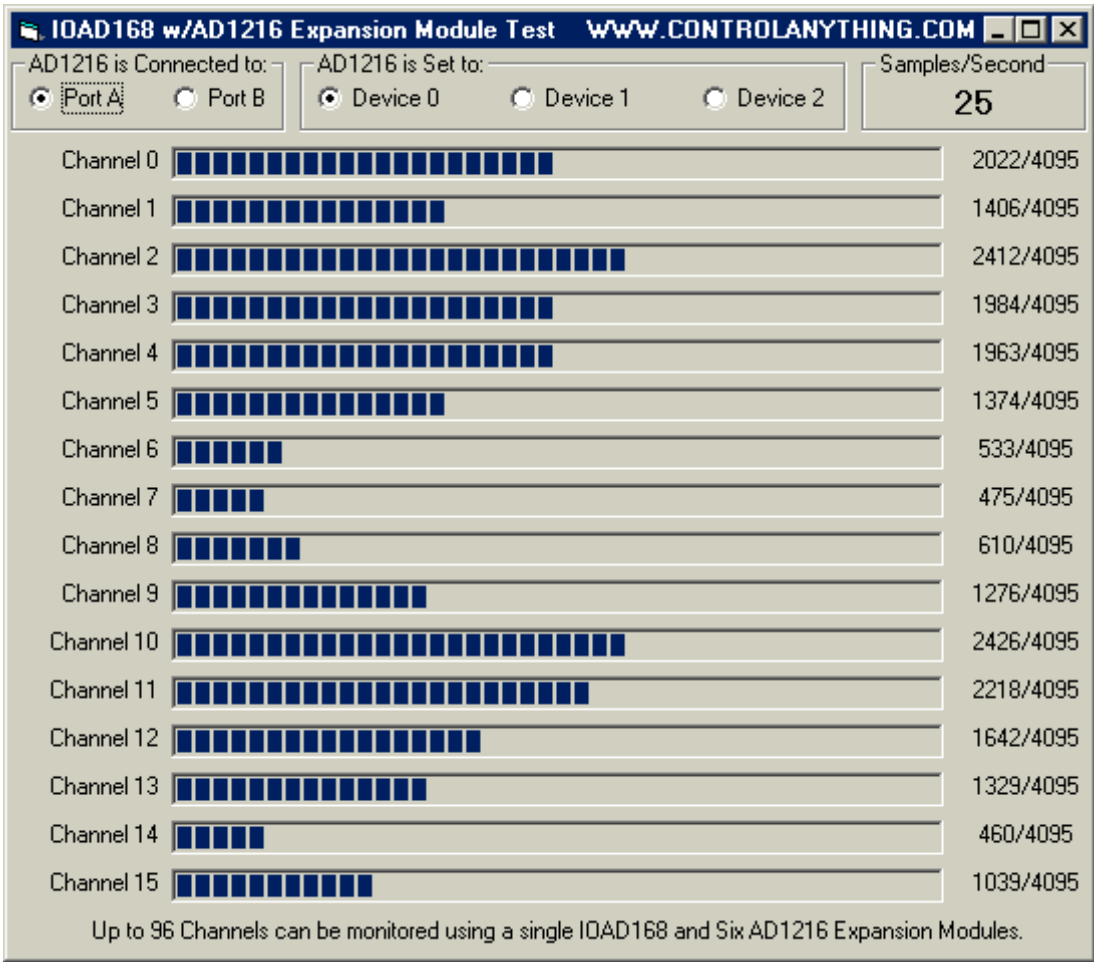
6, 14, 0-2, 0-15 Get 12-Bit Analog on Port B
This command returns a 12-Bit analog value from Channel 0-15, Defined as Device 0-2 on a AD1216 Connected to Port B.

Sample Code: AD1216

```
Public Function AD1216A(Device,Channel)
   MSComm1.Output = Chr$(254)     'Enter Command Mode
   MSComm1.Output = Chr$(6)       'Get Port Byte
   MSComm1.Output = Chr$(13)      'from AD1216 on Port A
   MSComm1.Output = Chr$(Device)  'AD1216 Device Number 0-2
   MSComm1.Output = Chr$(Channel) 'Analog Input Channel 0-15
   LSB = SerialIn
   MSB = SerialIn
   AD1216A = (MSB * 256) + LSB
End Function


Public Function AD1216B(Device,Channel)
   MSComm1.Output = Chr$(254)     'Enter Command Mode
   MSComm1.Output = Chr$(6)       'Get Port Byte
   MSComm1.Output = Chr$(14)      'from AD1216 on Port B
   MSComm1.Output = Chr$(Device)  'AD1216 Device Number 0-2
   MSComm1.Output = Chr$(Channel) 'Analog Input Channel 0-15
   LSB = SerialIn
   MSB = SerialIn
   AD1216B = (MSB * 256) + LSB
End Function

Public Function SerialIn()
        Timeout = 100
        Do
            Timeout = Timeout - 1
            If Timeout <= 0 Then
                SerialIn = -1
                Exit Function
            End If
            DoEvents
        Loop Until MSComm1.InBufferCount > 0
        SerialIn = Asc(MSComm1.Input)
End Function
```



We also provide a AD1216 demo program in the form of Visual Basic 6 Source Code at www.controlanything.com.

The IOAD168 Command Set

Introduction to the LCD and VF Character Display Command Set

The IOAD168 also includes several powerful commands for controlling up to three character LCD and VF displays. A character display may be attached to Port A, B, and/or C of the IOAD168. Each port has slightly different timing characteristics. Port C seems to be the most compatible, followed closely by port A. Port B character display routines operate slightly faster, and may suffer from incompatibility with some displays. Port B should NOT be used if you intend to use the IOAD168 as a terminal since Port B is the only port capable of reading a 16-button keypad.

Character displays are controlled in 4-bit mode using 7 of the 8 TTL outputs. The 8th TTL output can be used to software control the on/off status of a backlight.

It is important to understand that the CPU on the IOAD168 runs MUCH faster than a character display can keep up with. Therefore, some provisions have been taken to ensure the display is never overrun with text or commands. Most commands and data send ASCII character code 85 back to the user to indicate completion. This serves to slow communications just enough to ensure compatibility with just about every character display ever made up to 80 characters (such as 40x2 and 20x4 displays).

Commands are sent directly to the display at the hardware level, therefore, the user has complete control over the use of write-only commands to the character display. This guide will discuss the most command character display commands. Other commands may be issued using the functions we provide. A detailed data sheet on character displays should be obtained to use the advanced command set of character displays.

There are Seven character display commands currently supported by the IOAD168. Here is a brief overview of the command set. A detailed explanation will be provided on the following pages.

LCD Initialize
This command issues a general purpose initialization for all character displays. This command was written specifically to initialize a 16x2 display, but the init sequence is compatible with just about every character display produced. This command returns 85 back to the user when complete.

LCD Data
This command is used to send data to the display. Data is used for text and command parameters. This command returns 85 back to the user when complete.

LCD Command
This command is used to send a hardware command to the character display. Commands are used to position the cursor and other such functions. This command returns 85 back to the user when complete.

LCD Text Buffer
Due to speed limitations of a character display, text cannot be written directly to the screen from a serial output. Instead, text must be stored in a text buffer and then dumped to the display. This is managed entirely by the IOAD168 firmware. Simply fill the text buffer and terminate the command with ASCII character code 255 (more on this later). These data will then be rationed to the display from the text buffer. When the operation is complete, ASCII character 85 will be sent back to the user.

LCD LED Off
Turn Off Backlight LED. Initialization may be required after this command has be issued for some displays.

LCD LED On
Turn On Backlight LED. Initialization may be required after this command has be issued for some displays.

LCD Port
The IOAD168 has 3 parallel data ports, all of which are compatible with a character display. The LCD Port command is used to direct LCD commands to one of these data ports. By default, Port C is used. Port A may be selected by issuing this command with a parameter of 0. Port B may be selected by issuing this command with a parameter of 1. All subsequent LCD commands will be directed to the selected port. This allows an IOAD168 to control up to 3 character displays of different sizes at one time.

Character Display Related Commands

The IOAD168 also supports commands for controlling a keypad on Port B. This allows the IOAD168 to be used as a terminal interface to users in 256 different locations using a single serial port.

In some applications, it is necessary to direct key presses from the integrated keypad directly to the character display screen. The IOAD168 also supports a command that we call “Key-to-Screen”. Key-to-screen allows any key press on the keypad to be viewed directly on the character display.

In the case of PIN number and Pass Code requests, the IOAD168 can be configured to display an “X” (or any other character) to hide the key presses from other potential viewers.

Complete details on the Key-to-Screen features can be found in the Keypad section of this manual. The next few pages will help you utilize the character display capabilities of the IOAD168.

The IOAD168 Command Set

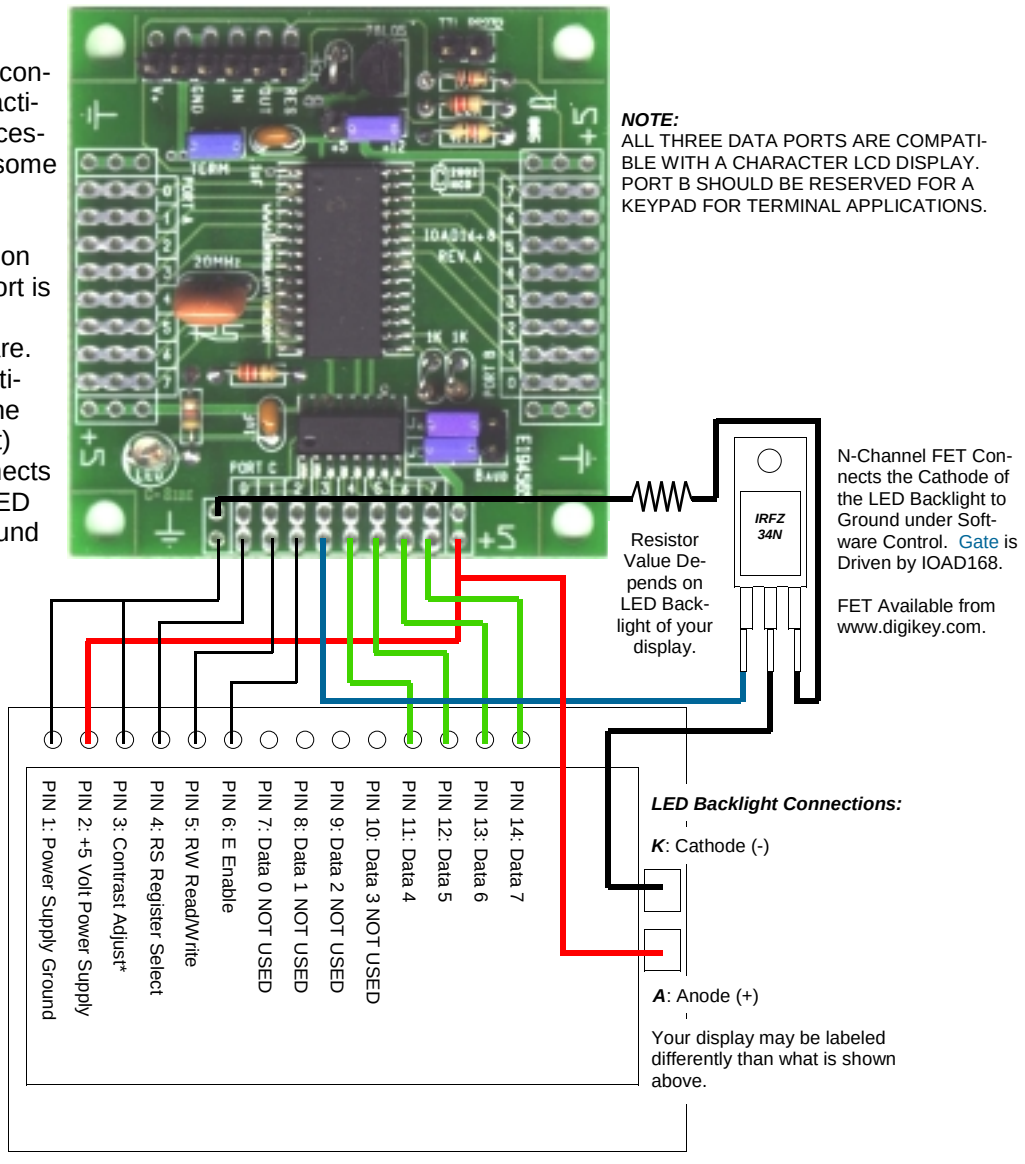
Connecting a Display to the IOAD168

The IOAD168 is capable of connecting to three different character display modules at one time. The user has complete software control of all display write functions as well as a software controlled backlight. The controller firmware has been tuned for compatibility with most character LCD and Vacuum Florescent displays.

Power for the display is derived from the IOAD168 controller. When controlling a Vacuum Florescent Display, you MUST provide a regulated +5 volt supply to the IOAD168 AND the power supply jumper below MUST be set to +5 volts to avoid damage to the on-board voltage regulator.

When the software controlled backlight is activated, it may be necessary to re-initialize some display models.

Data pin 3 (labeled on the board) of any port is used to control the backlight via software. When this line is activated, the gate of the FET (shown at right) goes high and connects the ground of the LED backlight to the ground of the controller.



NOTE:
ALL THREE DATA PORTS ARE COMPATIBLE WITH A CHARACTER LCD DISPLAY. PORT B SHOULD BE RESERVED FOR A KEYPAD FOR TERMINAL APPLICATIONS.

Resistor Value Depends on LED Backlight of your display.

N-Channel FET Connects the Cathode of the LED Backlight to Ground under Software Control. Gate is Driven by IOAD168.

FET Available from www.digikey.com.

LED Backlight Connections:

K: Cathode (-)

A: Anode (+)

Your display may be labeled differently than what is shown above.

The IOAD168 Command Set

IOAD168 Character Display Native Commands

When controlling a character LCD display with the IOAD168, it is important to realize there are two distinct command sets. There are commands that are we have built into the IOAD168 to control a character display (IOAD168 Native Commands), and there are commands built into the character display itself (LCD Native Commands).

The first type of commands we will discuss are the commands that we have designed to control a character display (IOAD168 Native Commands). There are seven commands used to control all character display write-only functions.

By default, all commands are routed to Port C. It is possible to route LCD commands to any of the data ports on the IOAD168 using the LCD Port command on the next page. Port B should not be used for character display functions if you intend to use a Keypad.

LCD Initialize

7, 0 LCD Initialize

This command is used to send an initialization sequence to the character display. While this command was written specifically to initialize a 16x2 display, this command is compatible with all displays we tested. When this command has finished, a flashing cursor should appear in the upper left corner of the display screen. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Initialize

```
Public Sub LCD_Init
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(0)            'Initialization Command
    GetData                             'Wait for Command to Finish
End Sub
```

LCD Data

7, 1, 0-255 LCD Data

This command is used to send data directly to the display hardware. This command can be used to display text on the screen, however, it is much more efficient to use the LCD Text Buffer command. This command is not used in most applications, but it is provided so just incase you need to adjust some display parameters. This command requires a parameter from 0-255 indicating the data that it is to be sent to the display. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Data

```
Public Sub LCD_Data(Data)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(1)            'LCD Data Command
    MSComm1.Output = Chr$(Data)         'Send Data 0-255
    GetData                             'Wait for Command to Finish
End Sub
```

LCD Command

7, 2, 0-255 LCD Command

This command is used to send a command directly to the display hardware. This command should be used to issue all commands native to the character display. Such commands allow you to be hide and position the cursor, program your own fonts, and clear the display. This command will be demonstrated on the following pages. This command requires a parameter from 0-255 indicating the command that it is to be sent to the display. Misuse of this command can cause unusual and erratic behavior. A complete set of commands is found in your LCD data sheet. Only the most common commands will be discussed in this manual. This command sends ASCII character code 85 back to the user indicating the command has finished execution.

Sample Code: LCD Command

```
Public Sub LCD_Command(Command)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(2)            'LCD Hardware Command
    MSComm1.Output = Chr$(Command)      'Send Command 0-255
    GetData                             'Wait for Command to Finish
End Sub
```

Sample Code: Wait for Command to Finish

```
Public Function GetData
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetData = Asc(MSComm1.Input)        'Get Status from Controller
```

The IOAD168 Command Set

LCD Buffer

7, 3, Text$, 255 LCD Buffer (64 Byte Text Buffer)
The IOAD168 has a very powerful text buffer built in that is very easy to use. A text buffer is required because the processor on the IOAD168 easily outruns the capabilities of a character display. Due to these speed limitations, text is stored in a buffer and slowly rationed to the display. To use the text buffer command, simply send: 254, 7, 3, "Text that you want to send.", 255. The 255 is a "Termination" command. It is used to signal the end of your text to the controller. Up to 64 characters can be stored in the text buffer. If the text buffer is filled, the display will update with the current contents of the text buffer and ignore excess data. Once text data has been written to the display, this command will send ASCII character code 85 back to the user indicating completion of this command.

Sample Code: LCD Buffer

```
Public Sub LCD_Buffer
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(3)            'Text Buffer Command
    MSComm1.Output = "Hello World"      'Text to Display
    MSComm1.Output = Chr$(255)          'Terminate and Display
    GetData                             'Wait for Command to Finish
End Sub
```

Backlight OFF

7, 4 LED Off
Turns off the backlight LED. Some displays need to be re-initialized after this command has finished execution. By default, the backlight LED is Off.

Sample Code: Backlight Off

```
Public Sub LCD_Backlight_Off
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(4)            'LED Backlight Off Command
End Sub
```

Backlight ON

7, 5 LED On
Turns on the backlight LED. Some displays need to be re-initialized after this command has finished execution.

Sample Code: Backlight On

```
Public Sub LCD_Backlight_On
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(5)            'LED Backlight On Command
End Sub
```

Routing Commands to IOAD168 Data Ports

7, 6, 0-2 LCD Port
It is possible to attach up to 3 character displays to a single IOAD168. One display is allowed per data port (A, B, or C). This command is used to route all subsequent LCD commands to the selected port. By default, LCD commands are sent to Port C.

You can route LCD commands to Port A using 254, 7, 6, 0.
All subsequent LCD commands will be sent to Port A.
You can route LCD commands to Port B using 254, 7, 6, 1.
All subsequent LCD commands will be sent to Port B.
You can route LCD commands to Port C using 254, 7, 6, 2.
All subsequent LCD commands will be sent to Port C.

Sample Code: Command Port Selection

```
Public Sub LCD_Port(Port)
    MSComm1.Output = Chr$(254)          'Enter Command Mode
    MSComm1.Output = Chr$(7)            'LCD Command
    MSComm1.Output = Chr$(6)            'LCD Hardware Port
    MSComm1.Output = Chr$(Port)         'Select Port A-C (0-2)
End Sub
```

Sample Code: Wait for Command to Finish

```
Public Function GetData
    Do                                  'Wait for Device to Reply
        DoEvents                        'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0     'If the Device Replies
    GetData = Asc(MSComm1.Input)        'Get Status from Controller
    Debug.Print GetData                 'Display in Immediate Window
End Sub
```

The IOAD168 Command Set

Native LCD Commands

A character display has its own built-in commands used to control cursor position, cursor appearance, fonts, and much more. Here are a few commands that can be sent to control these display functions. Other commands can be sent, please consult the manual for your character display for a complete set of commands and usage. The IOAD168 does not support any commands that read data from the display.

Basic HD44780 Command Set:

Command	Description
1	Clear Screen
2	Home (move cursor to top/left character position)
8	Blank the display (without clearing)
12	Make cursor invisible
12	Restore the display (with cursor hidden)
14	Turn on visible underline cursor
15	Turn on visible blinking-block cursor
16	Move cursor one character left
20	Move cursor one character right
24	Scroll display one character left (all lines)
28	Scroll display one character right (all lines)
64 + addr	Set pointer in character-generator RAM (CG RAM address)
128 + addr	Set cursor position (DDRAM address)

Sample Code: Native LCD Commands

```
Public Sub Clear_Screen()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(1)    'Clear Screen Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Home_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(2)    'Home Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Blank_Display()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(8)    'Blank Display Command (Does Not Clear)
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Invisible_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(12)   'Invisible Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Restore_Display()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(12)   'Restore Display Command (Cursor Invisible)
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Underline_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(14)   'Underline Cursor
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Block_Cursor()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(15)   'Block Cursor Command
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Cursor_Left()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(16)   'Move Cursor Left
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Cursor_Right()
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(7)    'LCD Function
    MSComm1.Output = Chr$(2)    'Send Command to LCD
    MSComm1.Output = Chr$(20)   'Move Cursor Right
    GetStatus                   'Wait for Command to Finish
End Sub

Public Sub Set_Cursor_Position(Pos)
    MSComm1.Output = Chr$(254)      'Enter Command Mode
    MSComm1.Output = Chr$(7)        'LCD Function
    MSComm1.Output = Chr$(2)        'Send Command to LCD
    MSComm1.Output = Chr$(128+Pos)  'Set Cursor Position (0-127)
    GetStatus                       'Wait for Command to Finish
End Sub
```

The IOAD168 Command Set

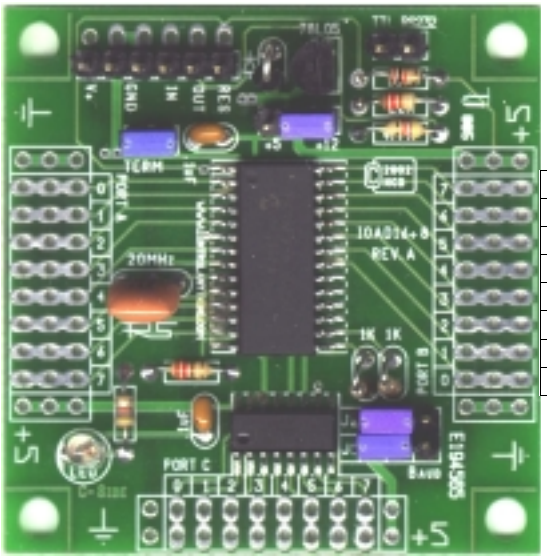
Keypad Encoder

The IOAD168 incorporates a very powerful programmable keypad encoder function. Any ASCII value from 0 to 255 can be assigned to each of 16 keys. See page 8 for assigning new values for each of the keys on the keypad.

Connecting a Keypad to the IOAD168 is very easy. All that is required is a suitable keypad (Matrix Type up to 16 Keys), and a Pull-Down Resistor Network (1K to 4.7K is suitable, 3.9K shown below).

The IOAD168 incorporates scan-based encoding algorithm to determine which keys are pressed. For this reason, 4 data bits are TTL level inputs and 4 are TTL level outputs. The encoder function detects an intersection between input and outputs and assigns a numeric value from 0-15. This numeric value is then cross referenced with the user-defined character set and sent out the serial port of the IOAD168.

A keypad can only be attached to Port B of the IOAD168. The photo below shows how to connect a keypad to the IOAD168.



Row A	Output
Row B	Output
Col 1	Input
Col 2	Input
Col 3	Input
Col 4	Input
Row D	Output
Row C	Output

Part Numbers:
The IOAD168 is compatible with matrix type keypads from many manufacturers. We used the Storm 700 and 900 series keypads from www.digikey.com for development. Part number MGR1113-ND. Part number 7731011102-ND is also a suitable 1K 10-Pin 9-Resistor network.

Keypad Functions

8, 0-127 Keypad
The Keypad function is used to acquire key presses from the user. This command requires a parameter from 0-127 indicating the number of key presses you would like the user to enter.

9, 0-127 Keypad to Screen
This command is the same as above, except that key presses are directed to a character display connected to Port A or Port C of the IOAD168. The keys that are stored in the controller (see page 8) will appear on the character display screen.

9,128-255 Keypad Hidden to Screen
This command is the same as above, except the key presses are not displayed on the screen. Rather, the hidden PIN character is used to acknowledge key presses from the user. The parameter value of (128-255) - 127 indicates the number of key presses. To acquire one key press from the user, send the command: 254, 9, 128. By default, the requested character will appear as an "X" on the character display screen. To acquire 4 key presses, send the command: 254, 9, 131. Four X's will appear on the screen as the key is pressed. The user will be sent the actual character value assigned to the keys (assignment is shown on page 8 of this manual).

Sample Code: Keypad Functions

```
Public Sub Keypad(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(8)    'Keypad Function
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (0-127)
    For N = 1 to Keys           'Display Key presses from User
       GetStatus
    Next N
End Sub

Public Sub KeyToScreen(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(9)    'Keypad Function Keys Displayed on LCD
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (0-127)
    For N = 1 to Keys           'Display Key presses from User
       GetStatus
    Next N
End Sub

Public Sub KeyToScreenHidden(Keys)
    MSComm1.Output = Chr$(254)  'Enter Command Mode
    MSComm1.Output = Chr$(9)    'Keypad Function Keys Displayed on LCD
    MSComm1.Output = Chr$(Keys) 'Get Key Presses (128-255)
    For N = 1 to Keys-127       'Display Key presses from User
       GetStatus
    Next N
End Sub
```